n reinforce algorithm(policy gradient), the gradient is calculated at end of trajectory. Now the formula for loss calculation ivolved -log\_prob\*discounted\_reward at every time step. But if we have episode where reward is only obtained at end eg. chess, them how is dicounted\_reward calculted per time step for above loss?

ChatGPT

In the Reinforce algorithm (a type of policy gradient method), the key idea is to adjust the policy parameters in the direction that maximizes the expected reward. When rewards are sparse and only received at the end of an episode, such as in chess, the calculation of the discounted reward at each time step can still be performed. Here’s how it works:

**Discounted Rewards in Sparse Reward Scenarios**

In environments where the reward is only received at the end of an episode, the discounted reward GtG\_tGt​ for every time step ttt can be computed as follows:

1. **Complete Episode Reward**: In such scenarios, the entire episode reward is received at the final step. Let's denote this final reward as RTR\_TRT​, where TTT is the time step at which the episode ends.
2. **Discounted Reward Calculation**:
   * For each time step ttt, the reward-to-go (or the return) is the sum of discounted rewards from time ttt to the end of the episode.
   * Since the reward is only received at the end, the reward-to-go for each time step is the discounted value of the final reward.

The formula for the discounted reward GtG\_tGt​ at each time step ttt becomes:

Gt=RT⋅γT−tG\_t = R\_T \cdot \gamma^{T-t}Gt​=RT​⋅γT−t

Here:

* + γ\gammaγ is the discount factor (usually between 0 and 1).
  + TTT is the final time step of the episode.
  + ttt is the current time step.

**Loss Calculation**

Given the above method for calculating discounted rewards, the loss at each time step ttt can be calculated using the policy gradient loss formula:

L(θ)=−log⁡πθ(at∣st)⋅GtL(\theta) = - \log \pi\_\theta(a\_t | s\_t) \cdot G\_tL(θ)=−logπθ​(at​∣st​)⋅Gt​

For each time step ttt, substitute GtG\_tGt​ with RT⋅γT−tR\_T \cdot \gamma^{T-t}RT​⋅γT−t:

L(θ)=−log⁡πθ(at∣st)⋅RT⋅γT−tL(\theta) = - \log \pi\_\theta(a\_t | s\_t) \cdot R\_T \cdot \gamma^{T-t}L(θ)=−logπθ​(at​∣st​)⋅RT​⋅γT−t

**Summary of Steps**

1. **Run an Episode**: Collect a trajectory by running the policy in the environment until the episode ends.
2. **Compute the Final Reward**: Obtain the reward RTR\_TRT​ at the end of the episode.
3. **Calculate Discounted Rewards**:
   * For each time step ttt in the episode, calculate the discounted reward as Gt=RT⋅γT−tG\_t = R\_T \cdot \gamma^{T-t}Gt​=RT​⋅γT−t.
4. **Compute Loss**:
   * For each time step ttt, compute the loss as L(θ)=−log⁡πθ(at∣st)⋅RT⋅γT−tL(\theta) = - \log \pi\_\theta(a\_t | s\_t) \cdot R\_T \cdot \gamma^{T-t}L(θ)=−logπθ​(at​∣st​)⋅RT​⋅γT−t.
5. **Update Policy**:
   * Use the calculated loss to perform a gradient update on the policy parameters.

This approach allows the policy gradient method to handle environments with sparse rewards effectively. Even though rewards are sparse, the policy still gets meaningful gradient information from the discounted rewards at each time step, guiding the policy towards actions that are likely to lead to high rewards in the future.

4o

say final reward is 10 and the episode length is 3 can you calculte the discounted rewards and show me

ChatGPT

Sure! Let's calculate the discounted rewards for an episode where the final reward is 10 and the episode length is 3. We'll use a discount factor γ\gammaγ to compute the discounted rewards at each time step.

**Parameters**

* Final reward RT=10R\_T = 10RT​=10
* Episode length T=3T = 3T=3
* Discount factor γ\gammaγ (we'll use a common value of 0.99 for this example)

**Discounted Rewards Calculation**

We need to calculate the discounted reward GtG\_tGt​ for each time step ttt using the formula:

Gt=RT⋅γT−tG\_t = R\_T \cdot \gamma^{T-t}Gt​=RT​⋅γT−t

Here are the time steps and corresponding calculations:

1. **Time step t=0t = 0t=0**

G0=10⋅0.993−0=10⋅0.993=10⋅0.970299=9.70299G\_0 = 10 \cdot 0.99^{3-0} = 10 \cdot 0.99^3 = 10 \cdot 0.970299 = 9.70299G0​=10⋅0.993−0=10⋅0.993=10⋅0.970299=9.70299

1. **Time step t=1t = 1t=1**

G1=10⋅0.993−1=10⋅0.992=10⋅0.9801=9.801G\_1 = 10 \cdot 0.99^{3-1} = 10 \cdot 0.99^2 = 10 \cdot 0.9801 = 9.801G1​=10⋅0.993−1=10⋅0.992=10⋅0.9801=9.801

1. **Time step t=2t = 2t=2**

G2=10⋅0.993−2=10⋅0.991=10⋅0.99=9.9G\_2 = 10 \cdot 0.99^{3-2} = 10 \cdot 0.99^1 = 10 \cdot 0.99 = 9.9G2​=10⋅0.993−2=10⋅0.991=10⋅0.99=9.9

So, the discounted rewards for each time step are:

* G0=9.70299G\_0 = 9.70299G0​=9.70299
* G1=9.801G\_1 = 9.801G1​=9.801
* G2=9.9G\_2 = 9.9G2​=9.9

**Summary**

For an episode of length 3 with a final reward of 10 and a discount factor of 0.99, the discounted rewards at each time step are:

* At t=0t = 0t=0: 9.702999.702999.70299
* At t=1t = 1t=1: 9.8019.8019.801
* At t=2t = 2t=2: 9.99.99.9

These values can then be used in the policy gradient loss calculation.

4o

now lets say I have 100 time steps and receive reward in every 5th time step then how will be calculate reward for other time steps?

ChatGPT

If you receive rewards every 5th time step in an episode of 100 time steps, you can calculate the discounted rewards for each time step by summing up the discounted future rewards. Here's how you can do it:

**Parameters**

* Episode length T=100T = 100T=100
* Rewards are received every 5th time step: R5,R10,R15,…,R100R\_5, R\_{10}, R\_{15}, \ldots, R\_{100}R5​,R10​,R15​,…,R100​
* Let's assume the reward at each of these time steps is RRR (constant for simplicity, but this can be adjusted to different values if the rewards vary)
* Discount factor γ\gammaγ (we'll use a common value of 0.99 for this example)

**Discounted Rewards Calculation**

For each time step ttt, the discounted reward GtG\_tGt​ is the sum of all future rewards, each discounted appropriately. The formula for GtG\_tGt​ at each time step ttt is:

Gt=∑k=t+1Tγk−tRkG\_t = \sum\_{k=t+1}^{T} \gamma^{k-t} R\_kGt​=∑k=t+1T​γk−tRk​

where RkR\_kRk​ is the reward received at time step kkk.

**Example Calculation**

Let's assume a constant reward RRR of 10 at each 5th time step.

**Steps**

1. Initialize an array G of length 100 to store the discounted rewards for each time step.
2. For each time step ttt, sum the discounted future rewards.

**Calculating Discounted Rewards**

1. **Time step t=0t = 0t=0**:

G0=10⋅γ5+10⋅γ10+10⋅γ15+…+10⋅γ100G\_0 = 10 \cdot \gamma^5 + 10 \cdot \gamma^{10} + 10 \cdot \gamma^{15} + \ldots + 10 \cdot \gamma^{100}G0​=10⋅γ5+10⋅γ10+10⋅γ15+…+10⋅γ100

In general, for t=0t = 0t=0, it sums up all rewards received in the future, each discounted by the number of steps between ttt and the reward time step.

1. **General Time Step ttt**: For any ttt, we only sum up the future rewards. For instance, at t=6t = 6t=6:

G6=10⋅γ9+10⋅γ14+…+10⋅γ94G\_6 = 10 \cdot \gamma^9 + 10 \cdot \gamma^{14} + \ldots + 10 \cdot \gamma^{94}G6​=10⋅γ9+10⋅γ14+…+10⋅γ94

Let's break down the code to compute this:

python

Copy code

import numpy as np

# Parameters

T = 100

R = 10

gamma = 0.99

# Initialize the rewards array

rewards = np.zeros(T)

rewards[4::5] = R # Set rewards at every 5th time step

# Initialize the discounted rewards array

discounted\_rewards = np.zeros(T)

# Calculate discounted rewards

for t in range(T):

G\_t = 0

for k in range(t, T):

G\_t += rewards[k] \* (gamma \*\* (k - t))

discounted\_rewards[t] = G\_t

discounted\_rewards

**Example Result**

The result discounted\_rewards will contain the discounted reward for each time step. Here is a small excerpt of what it might look like (values approximated):

* G0G\_0G0​: sum of 10⋅γ5,10⋅γ10,10⋅γ15,…,10⋅γ10010 \cdot \gamma^5, 10 \cdot \gamma^{10}, 10 \cdot \gamma^{15}, \ldots, 10 \cdot \gamma^{100}10⋅γ5,10⋅γ10,10⋅γ15,…,10⋅γ100
* G1G\_1G1​: sum of 10⋅γ4,10⋅γ9,10⋅γ14,…,10⋅γ9910 \cdot \gamma^4, 10 \cdot \gamma^9, 10 \cdot \gamma^{14}, \ldots, 10 \cdot \gamma^{99}10⋅γ4,10⋅γ9,10⋅γ14,…,10⋅γ99
* ...
* G95G\_{95}G95​: sum of 10⋅γ510 \cdot \gamma^510⋅γ5
* G96G\_{96}G96​: sum of 10⋅γ410 \cdot \gamma^410⋅γ4
* G97G\_{97}G97​: sum of 10⋅γ310 \cdot \gamma^310⋅γ3
* G98G\_{98}G98​: sum of 10⋅γ210 \cdot \gamma^210⋅γ2
* G99G\_{99}G99​: sum of 10⋅γ110 \cdot \gamma^110⋅γ1

By following this process, you can compute the discounted reward for each time step in the episode, accounting for rewards that are only received every 5th time step.

4o

are the above values of discounted rewards just estimates?

ChatGPT

The values of discounted rewards calculated using the method described are not estimates; they are the precise discounted rewards given the provided parameters. Here's a detailed breakdown of how the values are calculated accurately.

**Explanation**

Given an episode length T=100T = 100T=100, where rewards are received every 5th time step with a reward value R=10R = 10R=10 and a discount factor γ=0.99\gamma = 0.99γ=0.99, we calculate the discounted reward GtG\_tGt​ for each time step ttt.

**Steps for Calculation**

1. **Initialize the Rewards Array**: Create an array rewards where rewards are placed at every 5th time step.
2. **Initialize the Discounted Rewards Array**: Create an array discounted\_rewards to store the cumulative discounted reward for each time step.
3. **Calculate Discounted Rewards**: For each time step ttt, sum the discounted future rewards.

Here’s the Python code to perform these calculations:

python

Copy code

import numpy as np

# Parameters

T = 100 # Episode length

R = 10 # Reward value at every 5th time step

gamma = 0.99 # Discount factor

# Initialize the rewards array

rewards = np.zeros(T)

rewards[4::5] = R # Set rewards at every 5th time step

# Initialize the discounted rewards array

discounted\_rewards = np.zeros(T)

# Calculate discounted rewards

for t in range(T):

G\_t = 0

for k in range(t, T):

G\_t += rewards[k] \* (gamma \*\* (k - t))

discounted\_rewards[t] = G\_t

discounted\_rewards

**Explanation of the Code**

1. **Rewards Array**:
   * rewards[4::5] = R sets the reward RRR at every 5th time step starting from index 4 (i.e., time steps 5, 10, 15, ..., 100).
2. **Discounted Rewards Calculation**:
   * For each time step ttt, the cumulative discounted reward G\_t is calculated by summing the future rewards from ttt to TTT, each discounted by the factor γ\gammaγ raised to the power of the difference in time steps (k−t)(k - t)(k−t).

**Detailed Calculation Example**

For instance, to manually verify a few calculations:

1. **Time step t=0t = 0t=0**:

G0=10⋅γ4+10⋅γ9+10⋅γ14+…+10⋅γ99G\_0 = 10 \cdot \gamma^4 + 10 \cdot \gamma^9 + 10 \cdot \gamma^{14} + \ldots + 10 \cdot \gamma^{99}G0​=10⋅γ4+10⋅γ9+10⋅γ14+…+10⋅γ99

1. **Time step t=1t = 1t=1**:

G1=10⋅γ3+10⋅γ8+10⋅γ13+…+10⋅γ98G\_1 = 10 \cdot \gamma^3 + 10 \cdot \gamma^8 + 10 \cdot \gamma^{13} + \ldots + 10 \cdot \gamma^{98}G1​=10⋅γ3+10⋅γ8+10⋅γ13+…+10⋅γ98

1. **Time step t=5t = 5t=5**:

G5=10⋅γ4+10⋅γ9+…+10⋅γ94G\_5 = 10 \cdot \gamma^4 + 10 \cdot \gamma^9 + \ldots + 10 \cdot \gamma^{94}G5​=10⋅γ4+10⋅γ9+…+10⋅γ94

Each GtG\_tGt​ is the exact sum of all future rewards discounted to the current time step ttt.

**Conclusion**

The calculated values in the discounted\_rewards array are precise and not estimates. They accurately represent the cumulative discounted future rewards for each time step based on the given reward structure and discount factor. This approach ensures that you get the exact values of discounted rewards required for policy gradient methods like REINFORCE.

4o